Optimizing Data Access With Visual Basic® 6.0 Part I

William R. Vaughn
Developer Trainer
Microsoft Technical Education
Microsoft Corporation


POWER UP
Windows DNA 2000
Readiness Conference
featuring SQL Server 2000

William R. Vaughn

Developer Trainer, Microsoft Corporation

Author:

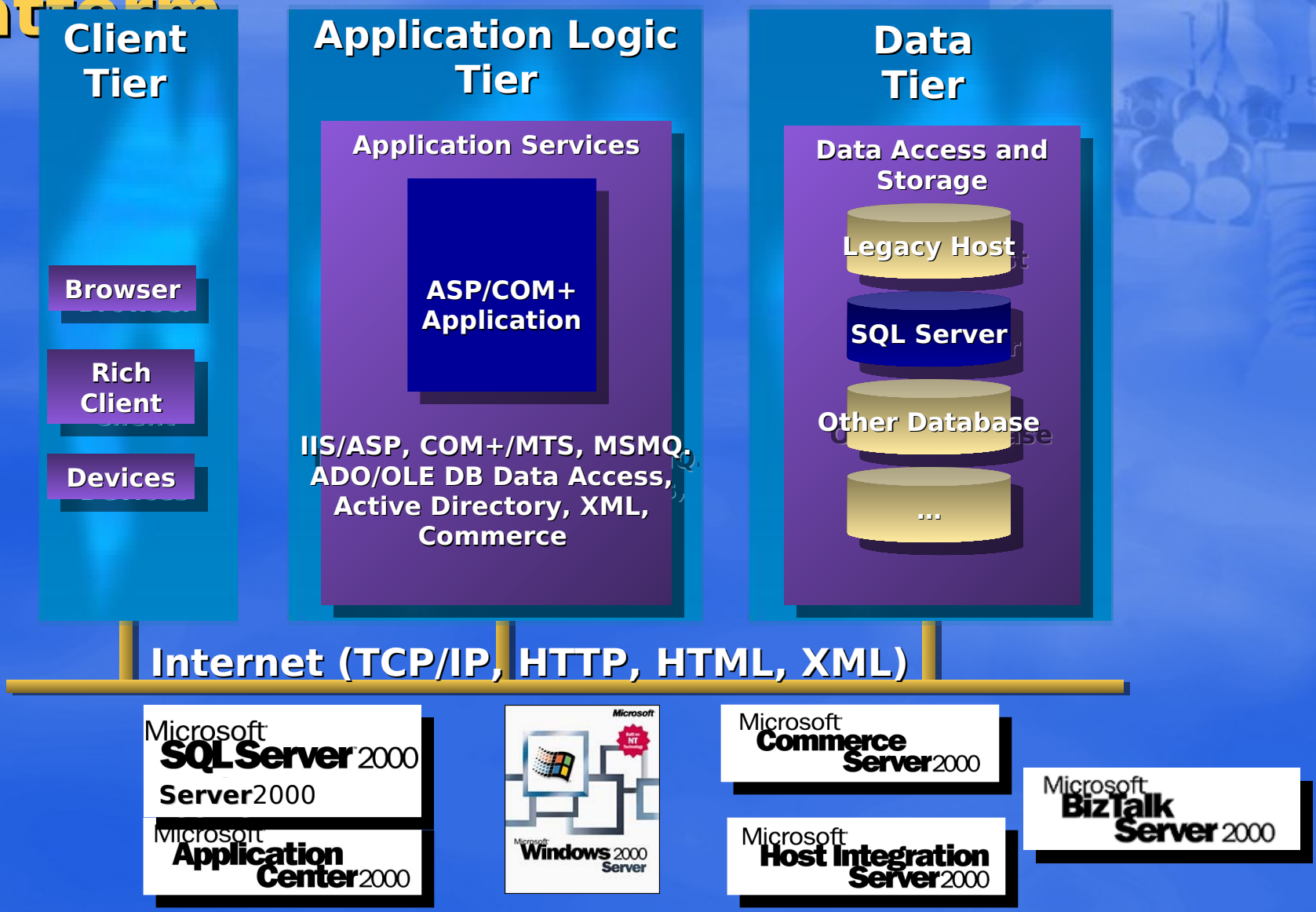
Hitchhiker's Guide to Visual Ba
SQL Server (now in 6 langua

billva@nwlink.com

www.betav.com

Windows DNA 2000

Next Generation Web Application Platform

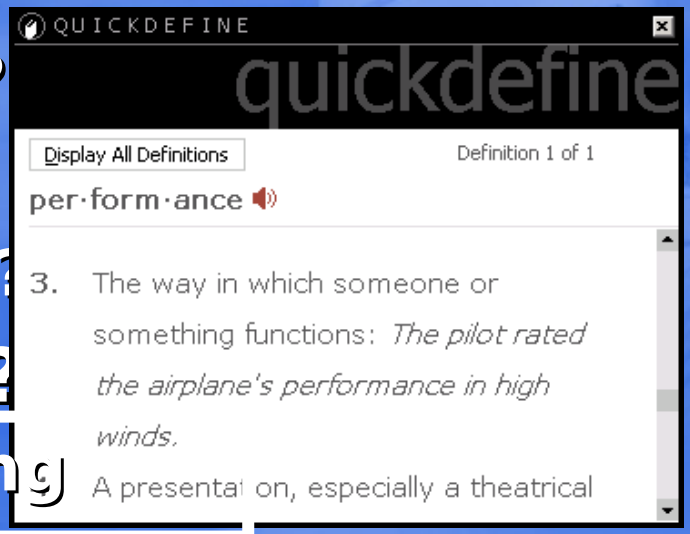


Optimizing Visual Basic Data Access Performance

- Performance
  - What does it mean?
  - How is it measured?
- Achieving performance goals
  - Through smarter designs
  - Through smarter coding
  - Through smarter users

Performance

- What does it mean?
  - Faster applications?
  - Faster development?
- How is it measured?
  - Fewer seconds waiting
  - More aggravated users
    - units/second
    - bugs
    - lines/day
    - lines of code…
    - satisfied customers…
    - repeat business



Smarter Designs

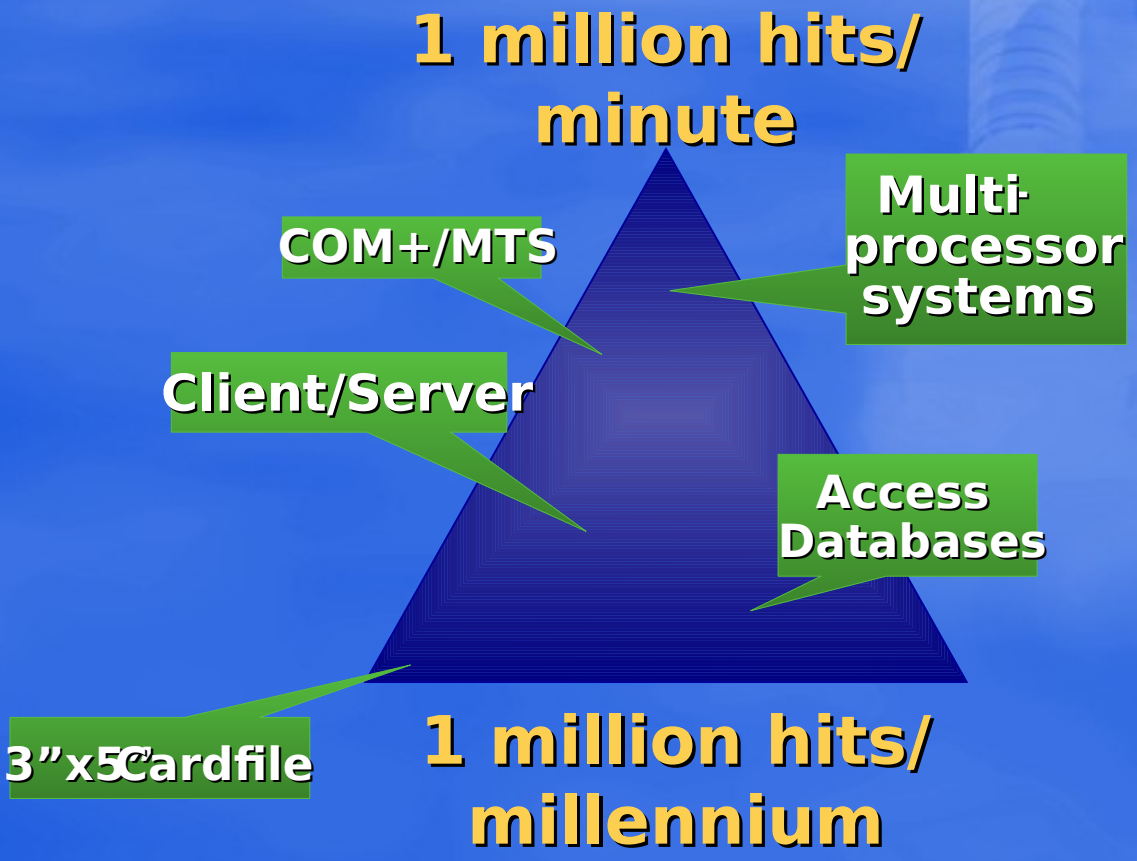
- Choose the “right” solution
  - For the customer
  - For the system(s)
  - For the developer(s)

Smarter Designs

- Smarter design strategies
- Smarter COM strategies
- Smarter coding strategies
- Smarter ADO strategies

Smarter Design Strategies

- Use the right solution in the right place



Smarter COM Strategies

- Recycle reusable objects
  - Command, connection objects
- Discard single-use objects quickly
  - Cleanup after yourself. Close unneeded…
    - Connection, recordset objects
  - Set unused objects to nothing
- Don’t use Dim xx as New yy
  - Use Dim xx as yy and Set xx = New yy

Smarter COM Strategies

- Coding ADO (or any) object variables
- Dim xx as new yy… the fallout:
  - Visual Basic adds the following to

```
If xx is Nothing then Set xx = New yy
```

- Use explicit set statement

```
Dim rs as Recordset
Set rs = New Recordset
```

Smarter COM Strategies

- ***Set objects* to Nothing**

```
rs.Close
Set rs = Nothing
```

- ***Properties* = Nothing**

```
rs.ActiveConnection = Nothing
```

Smarter COM Strategies

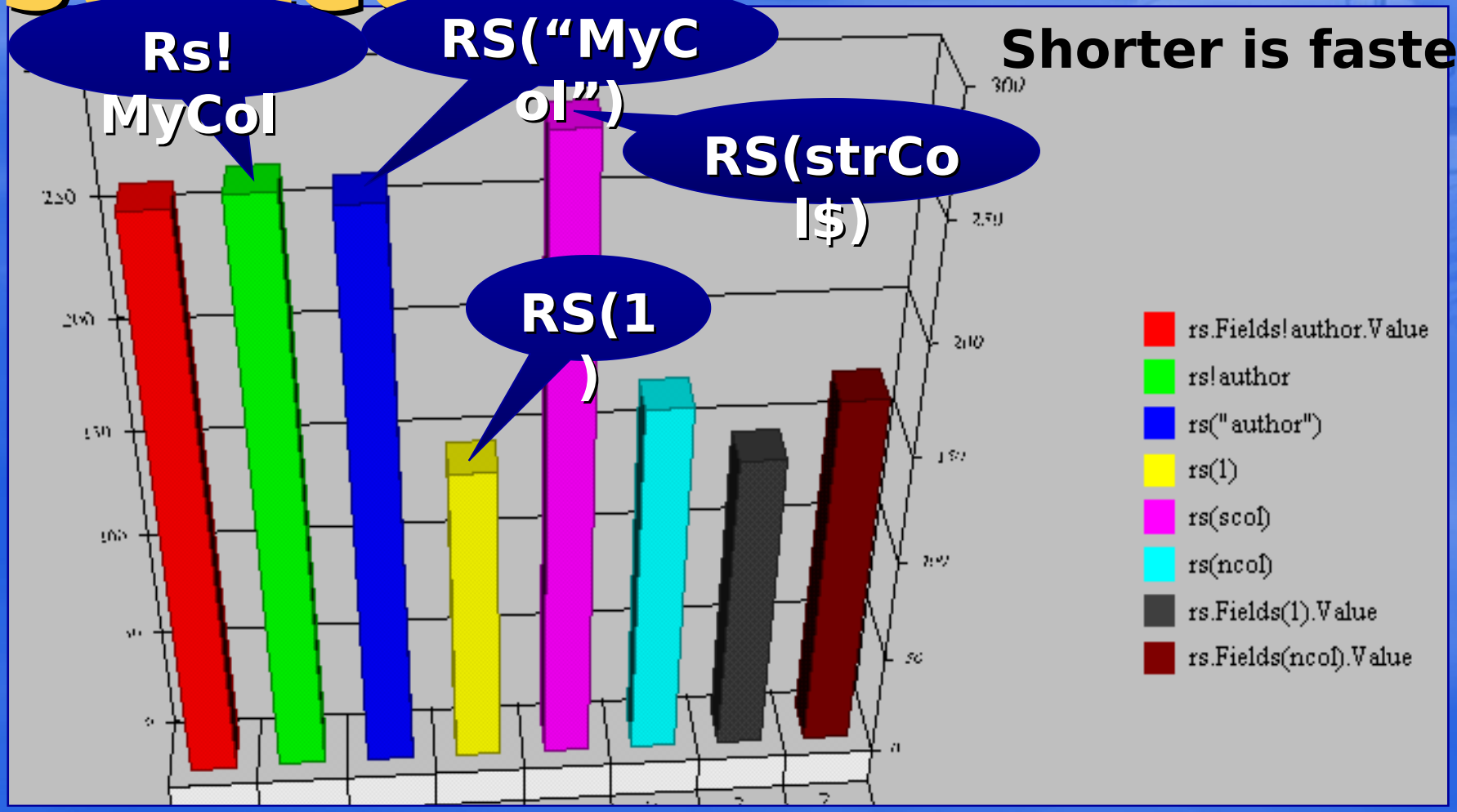
Fastest

```
Dim Rs as ADODB.Recordset       ' ambiguity

...

Rs(0)                           ' by ordinal

Rs.Fields(0).Value              ' explicitly

Rs.Fields(intF).Value           ' by variable

Rs(intF)                    ' by variable

Rs!MyFieldName              ' by name

Rs.Fields!Author.Value          ' Bang field
```

Speed

Slowest

Smarter COM Strategies



Smarter COM Strategies

- Enumerate Field and Parameter references

```
Enum enuFld
   ID
   Name
   Age
End Enum

Rs(enuFld.Age) = txtGetAge.Text
```

Smarter COM Strategies

- Use for each syntax
  - Shortcuts COM references
  - Makes code easier to read, debug

```
Dim errADO as ADODB.Error
For Each errADO in Cn.Errors
      Debug.print errADO.Description, _
            errADO.Number
Next
```

Smarter COM Strategies

- Use *with* syntax
  - Shortcuts COM references
  - Makes code easier to read, debug

```
With cmdGetAuthorByYear
   .CommandText = “Au65”
   .CommandType = adCmdStoredProc
   .ActiveConnection = cn
   .Refresh
End With
```

ADO: Smarter Coding

- Using Connection objects
- Using Command objects
- Using Recordset objects

Smarter Coding: Where?

- In ActiveX® Server Pages (ASP)
- In client-side code
- In the middle tier
- On the server(s)

Connection Strategies

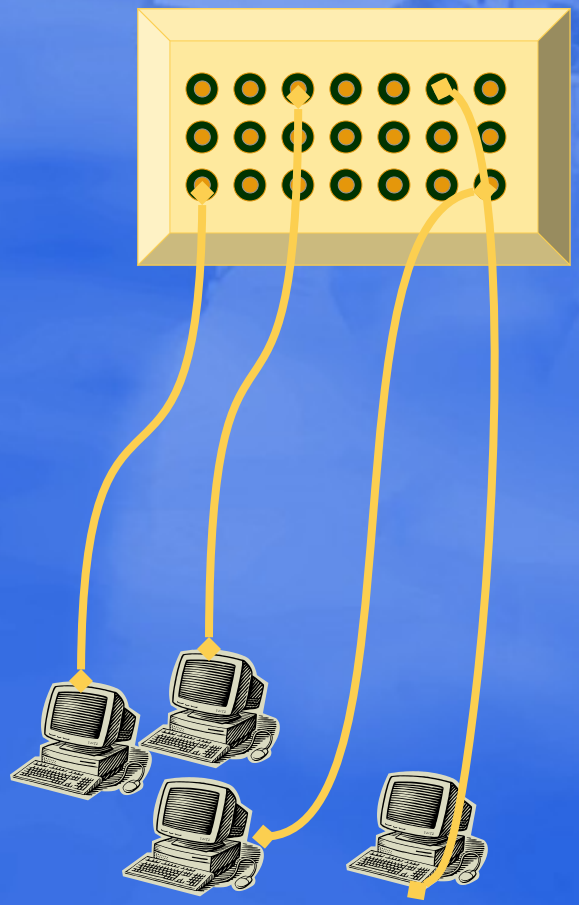
- Connect, query, populate…
  - Static: … and hold
  - JIT: … drop and reconnect as needed
  - Reuse: … and draw from pool
  - Dissociate: … discard and re-associate…
- Extract data from URL

Connection Strategies

- Don’t connect at all…
  - Use locally persisted Recordsets
  - Construct in-memory Recordset
    - Declare
    - Append fields
    - Open
    - Add, change, delete, update
    - Save

Connection Strategies

- Benefits
  - “Instant” connections
  - More total, fewer “active” users
  - Local Recordset persistence, creation
  - Scalability
  - Performance



Using Connection Objects

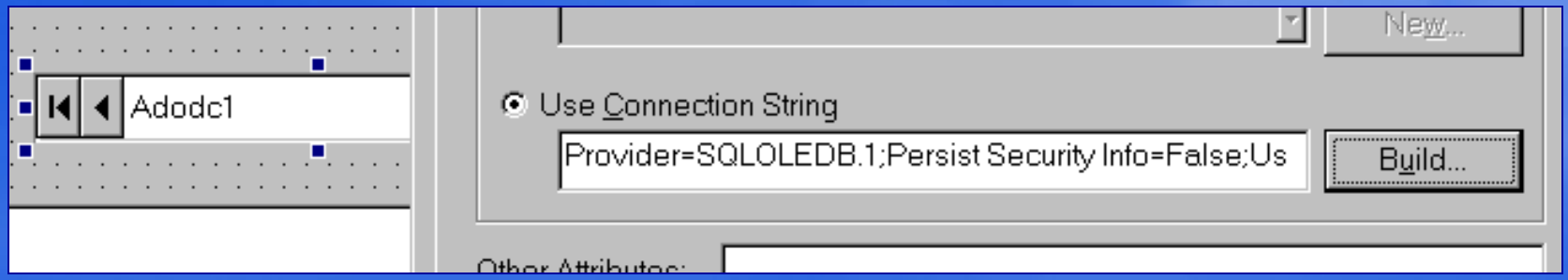
- ODBC versus OLE DB “native” providers
- DSNs versus UDL
- File versus registry-based
- Cached connections
- Hard-coding
  - Server name, IP, URL
  - User ID, password

Connection Strategies

- Be ready for “stuff” that happens
  - Include robust error managers
- Avoid tools, controls that
  - Deny control
  - Expose unwanted dialogs
- Watch out for residual state

Development Shortcuts

- Avoid trying to invent yourself…
- "Leverage" connection strings
  - From ADO Data Control
  - From known working code



Using Command Objects

- ***Required*** **for**
  - **Return status**
  - **Output parameters**
- **Only helpful for**
  - **Input parameters**
  - **Improved query performance (maybe)**
- **Often constructed automatically**

Constructing Parameters

- Leverage DataView Window
- Use one-line syntax:

```
With cmd
   .Parameters.Append .CreateParameter(…
```

- Construct Command once
- Re-use as needed

Smarter Commands

- Use stored procedures
- Reuse Command objects
- Return output parameters
- Leverage sp_executesql

Smarter Commands

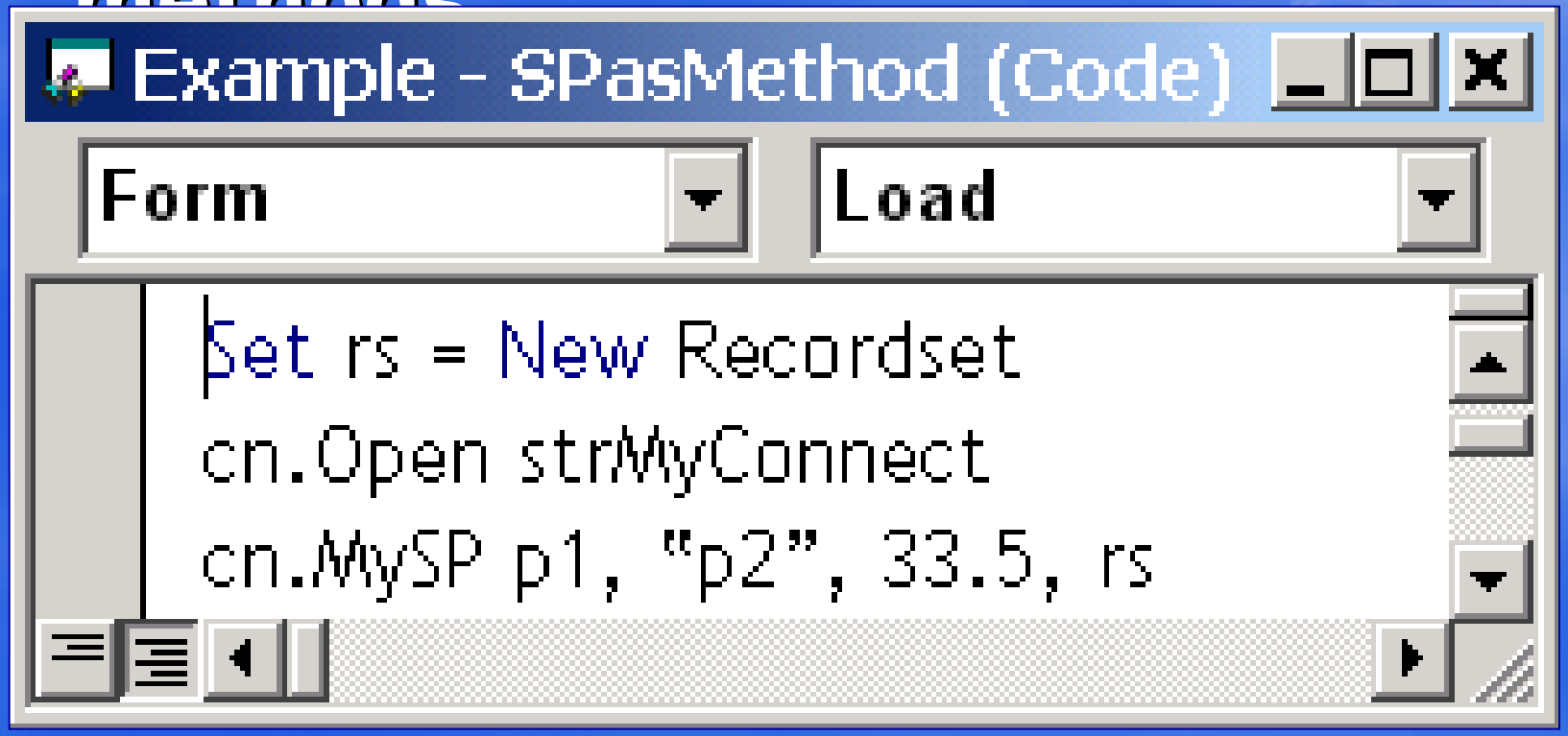
- Balance use of Refresh method
- Avoid use of Prepare property
- Observe Profiler, NetMon

Smarter Commands

- Execute Stored Procedures
  - Pre-compiled queries persisted in DB
  - Many stored procedures not updatable
- Accept
  - Parameters
- Return
  - Parameters, return status
  - more complex (multiple) result sets

Smarter Commands

- ADO lives for stored procedures
- All SPs exposed as Connection methods



Smarter Recordsets

- Required to handle
  - Rowsets from any source
  - Cursors—even “firehose”
  - Disjoint, persisted Recordset objects
- Not required to handle
  - Query execution
  - Output parameters
  - Return status
  - Update, delete, insert operations

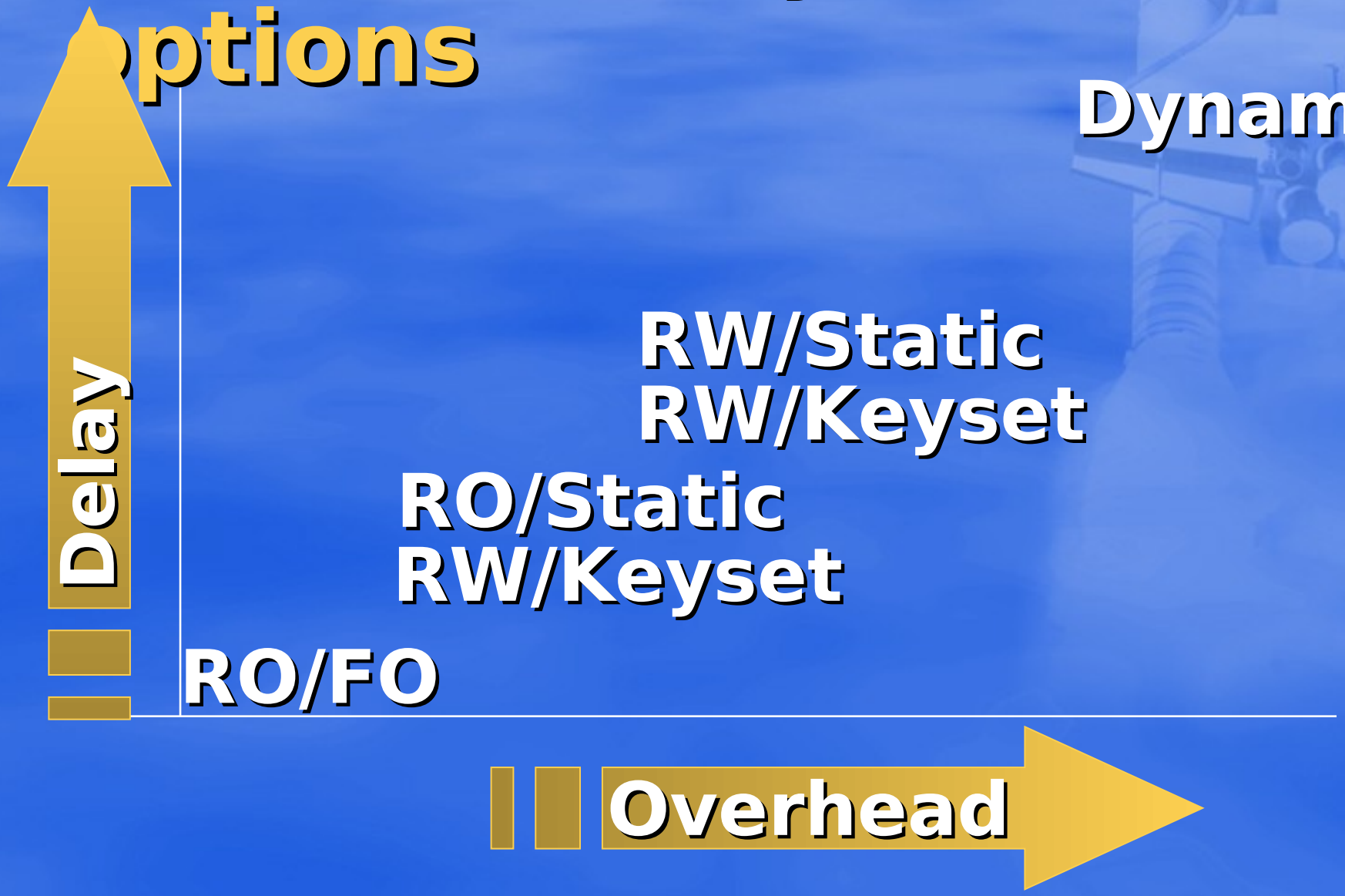
Smarter Recordsets

- Default cursor
  - Not really a cursor (RO/FO/CS=1)
  - Read-only, Forward only, CacheSize = 1
- Not needed for many queries
- Not universally supported

Smarter Recordsets

- Choose the lightest-weight mechanism
  - *Just* the columns needed (by name)
  - *Never* SELECT *
- Fetch only the rows needed (now)
  - Always include a WHERE clause
  - No more than a few dozen rows
- Populate quickly and release

ADO CursorType Options



Handling Recordsets

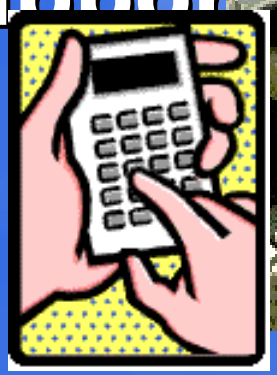
- Population
  - MoveLast (where supported)
  - MoveNext until RS.EOF
  - …or wait until ADO does it for you
- Upload
  - Do Until RS.EOF
  - GetRows—returns Variant array
  - GetString—returns “delimited” string

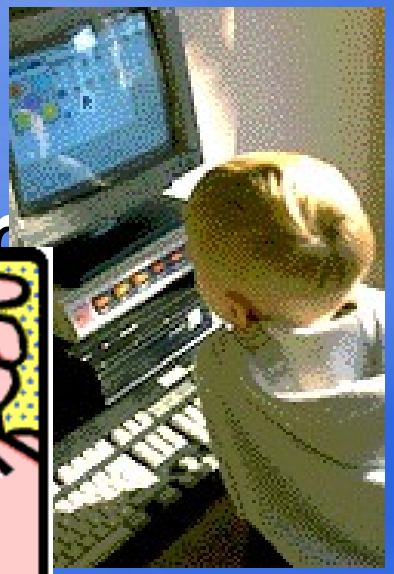
Passing Smart Data

- Disjoint Recordset
- Delimited string
- User-defined structure
- Variant array
- PropertyBag object
- MSMQ messages

Measuring Performance

- Connections made more quickly
- Query starts and completes faster
- Recordsets return answer faster
- More users/second suppor
- Things seem faster…





Measuring Tools

- GetTickCount
  - API-based counter
- SQL Profiler
  - SQL Server 7.0 trace utili
- SQL Query Analyzer
  - Check out “generated” queries
- NetMon
  - Network monitor

Smarter Users

- Better documentation
- Better training
- Better support
- Better attitude

Better Performance

- Both systems and developer performance
  - Achievable
  - Measurable
  - Cost-effective

POWER UP

Microsoft